**Assembly**

Fundamental unit of deployment

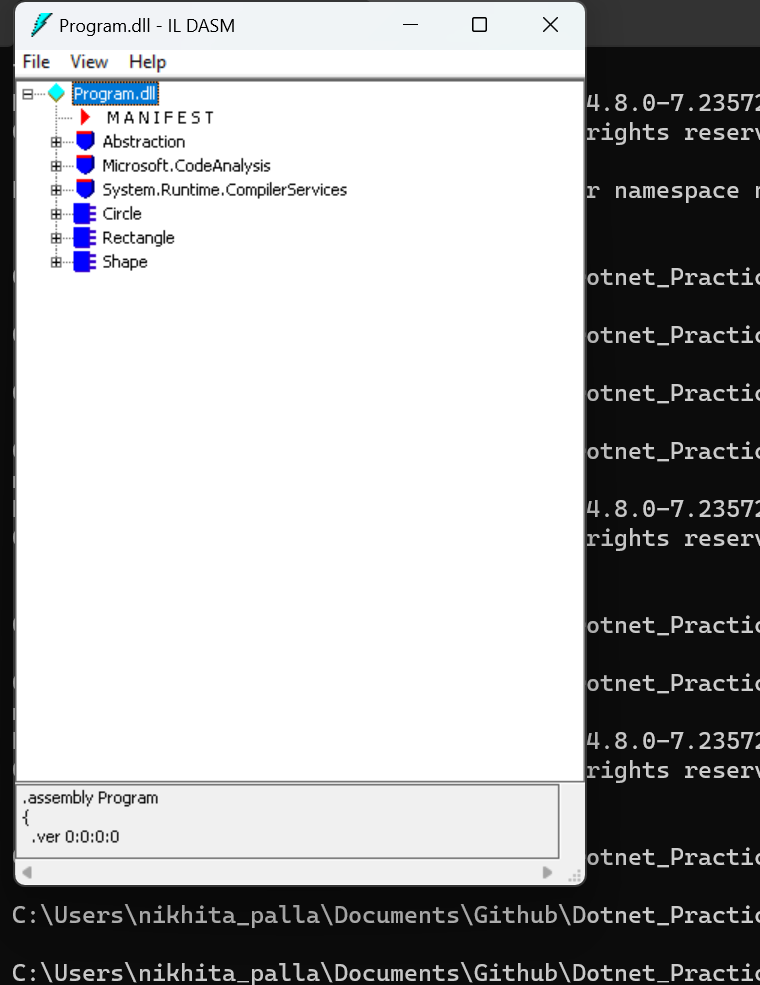
Pre-compiled chunk of dotnet code, which can run by CLR.

What is an assembly:

An assembly is a group of forms and assets that have been designed to interact with one another and provide a specific set of capabilities when used together.

The information required by the CLR, with which it needs to be conscious is present in the assembly.

Content of C# assembly



In dotnet assemblies are the basic building blocks for:

* Implementation
* Version Control
* Initialisation
* Safety privileages

Assembly cache

When .net framework is installed a new directory is added to the hard drive during the installation process. We store all the assembled files in this assembly cache.

Cache is divided into:

* Private : restricted files for applications.
* Global: All assemblies in the global cache must have different namespaces. The folder name have the assembly distinct identity.

Assembly has four parts:

* Assembly name
* Version number
* Cultural Information
* Public key token

There are two types of Assemblies :

1. Private Assemblies

Used for local application development.

Each application has its own private class of assemblies.

It is easy to deploy in the application as it kept local to the application and as a result there are no versioning conflicts with other applications.

If multiple applications wants to use the same assembly then we need to develop the assembly for each application separately as a result we duplicate the code and disk space usage is increased.

1. Shared Assemblies

Used by multiple applications.

All these shared assemblies are stored in GAC ( Global Assembly Cache)

GAC : a common location for shared assemblies on a computer.

Reduces code reusability and avoids code duplication.

But, if not deployed properly with appropriate steps there is a chance of version conflicts.

There are two types of Shared assemblies:

1. Strong-named Assemblies

Shared assemblies that are associated with a unique key pair providing unique identifier for the assembly.

Prevents versioning conflicts

Allows to add into the Gac

This is useful in scenarios where multiple versions of the same assembly coexist.

1. Satelite Assemblies

Type of shared assembly that contains localized resources (e.g., strings, images) for specific cultures or languages. Used to separate language-specific resources from the main application, enabling support for multiple languages.

Allows applications to be easily localized without modifying the main executable.

GAC:

Location : C:\Windows\Microsoft.NET\assembly

To ways to install assembly to GAC:

1. Simply drag and drop
2. GacUtil.exe (GAC Utility Tool)

Adding an assembly to GAC:

1. Create an assembly
2. Create a key pair

Strong named assembly can be saved into GAC.

In visual studio developer cmd :

sn.exe -k C:\MyStrongKeys.sk

Specify the below line in the assembly properties/assemblyinfo.cs ( at the end)

[assembly: AssemblyKeyFile("C:\MyStrongKeys.snk")]

1. Go to the assembly created .dll file

cd C:\Users\nikhita\_palla\Documents\Github\Dotnet\_Practice\C\_Sharp Basics Coding\Calculator\_Assembly\Calculator\_Assembly\bin\Debug

1. Adding to GAC

gacutil.exe -i Calculator\_Assembly.dll

1. Check in C:\Windows\Microsoft.NET\assembly\GAC\_MSIL if Calculator\_Assembly is added or not.
2. To uninstall the GAC added assembly

gacutil.exe -u Calculator\_Assembly.dll

1. Using The Gac assembly in program

Calculator\_Assembly.Class1 calculator = new Calculator\_Assembly.Class1();

Assemblies can be of various forms:

1. Exe

Executable.

Runs in its own space.  
we can run these .exe files by just opening it, as they have their own address spaces.

Can be directly run by the operating system.

Can include both application code and the resources needed for its execution.

Have a specific entry point like Main method in c#.

These are loaded when the application is launched, this is called static linking.

The exe files are standalone and encapsulate the entire application, so change in exe donot affect other applications.

To access a Program.exe file for Program.dll

csc Program.cs

To access the IL :

ildasm Program.exe

1. Dll

Dynamic link library

Doesn’t run in its own space needs and hoster to consume it.

Not meant to be directly executed. They are loaded at runtime and used in other applications.

The code if wanted to reuse, can be written using DLL and run in the .exe address space.

Promotes code reusability and modularity.

Donot have specific entry point.

DLL are loaded dynamically when needed, so these are called dynamic link libraries.

Dynamic linking allows sharing of the same copy of DLL in multiple applications.

As a DLL is used by multiple applications, change in one DLL can affect the other applications that are using it.

To access a Program.dll file for a Program.cs:

csc /target:library /out:Program.dll Program.cs

To access the Program.dll

ildasm Program.dll